(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2020-0142754
(43) 공개일자 2020년12월23일

(51) 국제특허분류(Int. Cl.)
G06F 21/62 (2013.01) *G06F 21/60* (2013.01)
(52) CPC특허분류
G06F 21/62 (2013.01)
G06F 21/602 (2013.01)
(21) 출원번호 10-2019-0070006
(22) 출원일자 2019년06월13일
심사청구일자 2019년06월13일

(71) 출원인
경희대학교 산학협력단
경기도 용인시 기흥구 덕영대로 1732 (서천동, 경희대학교 국제캠퍼스내)
(72) 발명자
조진성
경기도 수원시 영통구 봉영로 1620 대우월드마크 영통 101-1801
손희승
경기도 용인시 수지구 동천로 64 동천마을동문굿모닝힐5차아파트 501-2002
이기영
경기도 성남시 중원구 도촌남로 22 휴먼시아섬마을1단지아파트 113-404
(74) 대리인
특허법인지원

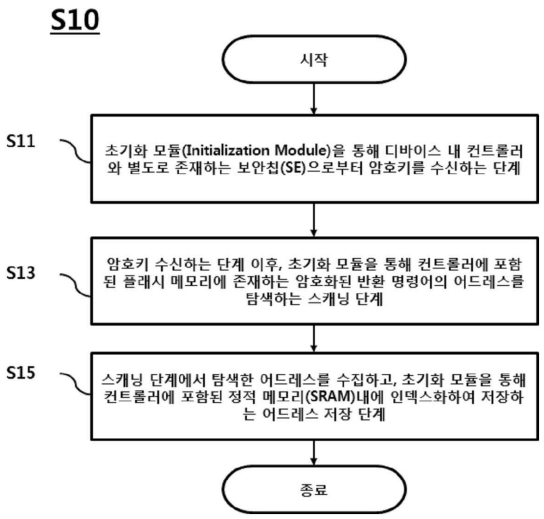
전체 청구항 수 : 총 7 항

(54) 발명의 명칭 **반환 명령어 난독화 방법 및 난독화된 반환 명령어 실행방법**

(57) 요 약

본 발명은 반환 명령어 난독화 방법 및 난독화된 반환 명령어 실행방법에 관한 것이다. 상기 방법은 컨트롤러가 수행하는 펌웨어 내 반환 명령어 난독화 방법에 있어서, 초기화(Initialization) 모듈을 통해 디바이스 내 상기 컨트롤러와 별도로 존재하는 보안칩(Secure Element: SE)으로부터 암호키를 수신하는 단계, 암호키 수신하는 단계 이후, 초기화 모듈(Initialization Module)을 통해 상기 컨트롤러에 포함된 플래시 메모리에 존재하는 암호화된 반환 명령어의 어드레스를 탐색하는 스캐닝 단계 및 스캐닝 단계에서 탐색한 어드레스를 수집하고, 초기화(Initialization) 모듈을 통해 컨트롤러에 포함된 정적 메모리(SRAM)내에 인덱스화하여 저장하는 어드레스 저장 단계를 포함할 수 있다.

대 표 도 - 도1



(52) CPC특허분류
G06F 2221/2125 (2013.01)

이 발명을 지원한 국가연구개발사업

과제고유번호	20182499
부처명	정보통신기획평가원
과제관리(전문)기관명	정보통신기술진흥센터
연구사업명	정보통신기술인력양성/SW중심대학지원사업
연구과제명	[2019 정부] SW중심대학(경희대)
기 여 율	1/1
과제수행기관명	경희대학교 산학협력단
연구기간	2017.03.01 ~ 2020.12.31

명 세 서

청구범위

청구항 1

컨트롤러가 수행하는 펌웨어 내 반환 명령어 난독화 방법에 있어서,

초기화(Initialization) 모듈을 통해 디바이스 내 상기 컨트롤러와 별도로 존재하는 보안칩(Secure Element: SE)으로부터 암호키를 수신하는 단계; 및

상기 암호키 수신하는 단계 이후, 상기 초기화(Initialization) 모듈을 통해 상기 컨트롤러에 포함된 플래시 메모리에 존재하는 암호화된 반환 명령어의 어드레스를 탐색하는 스캐닝 단계를 포함하는,

반환 명령어 난독화 방법.

청구항 2

제1항에 있어서,

상기 스캐닝 단계에서 탐색한 어드레스를 수집하고, 상기 초기화(Initialization) 모듈을 통해 상기 컨트롤러에 포함된 정적 메모리(SRAM)내에 인덱스화하여 저장하는 어드레스 저장 단계를 더 포함하는,

반환 명령어 난독화 방법.

청구항 3

제1항에 있어서,

상기 암호키를 수신하는 단계 이후,

상기 수신한 암호키를 상기 컨트롤러에 포함된 정적 메모리(SRAM)에 저장하는 단계를 더 포함하는,

반환 명령어 난독화 방법.

청구항 4

제1항에 있어서,

상기 초기화 모듈(Initialization Module)은,

코드 인스트루먼테이션 과정을 통해 소스 바이너리 파일에 삽입되며, 상기 플래시 메모리에 독립적으로 존재하는 모듈인,

반환 명령어 난독화 방법.

청구항 5

제1항 내지 제3항의 반환 명령어 난독화 방법에 의해 난독화된 반환 명령어를 컨트롤러에서 실행하는 방법에 있어서,

상기 플래시 메모리에 기저장된 펌웨어를 실행할 때, 정적 메모리(SRAM)에 저장된 암호키를 이용하여 상기 암호화된 반환 명령어를 복호화하고, 상기 복호화된 반환 명령어를 실행하는 실행(Runtime) 단계를 포함하는,

난독화된 반환 명령어 실행방법.

청구항 6

제5항에 있어서,

상기 실행(Runtime) 단계는,

상기 컨트롤러가 상기 암호화된 반환 명령어를 실행하기 이전에 반환 제어 흐름(Return Control Flow) 모듈을 실행하는 단계를 더 포함하는,

난독화된 반환 명령어 실행방법.

청구항 7

제6항에 있어서,

상기 반환 제어 흐름(Return Control Flow) 모듈은,

코드 인스트루먼테이션 과정을 통해 소스 바이너리 파일에 삽입되며, 상기 암호화된 반환 명령어를 복호화하여 실행하는 모듈이며,

상기 컨트롤러가 펌웨어 실행하여, 상기 반환 제어 흐름(Return Control Flow) 모듈이 호출신호를 받으면,

상기 정적 메모리(SRAM)에 저장된 어드레스를 로드하는 단계;

상기 암호화된 반환 명령어를 상기 암호키를 이용하여 복호화하며, 상기 복호화된 명령어를 정적 메모리(SRAM)에 저장하는 단계; 및

상기 정적 메모리(SRAM)에 저장된 복호화된 명령어를 실행하는 단계를 포함하는,

난독화된 반환 명령어 실행방법.

발명의 설명

기 술 분 야

[0001] 본 발명은 반환 명령어 난독화 방법 및 난독화된 반환 명령어 실행방법에 관한 것으로, 더욱 상세하게는 초기화(Initialization) 모듈을 이용하여 반환 명령어를 난독화하는 방법 및 반환 제어 흐름(Return Control Flow) 모듈을 이용하여 난독화된 반환 명령어를 실행하는 방법에 관한 것이다.

배 경 기 술

[0002] 사용자 중심의 지능형 서비스 제공의 선두주자로써 IoT 기술이 급성장 하고 있다. IoT 디바이스는 센서 네트워크 기술을 접목시켜 데이터를 수집하고 이를 기반으로 사용자에게 서비스를 제공하기 때문에 다양한 분야에서 활용되고 있으며 디바이스의 개체 수 또한 급증하고 있다. 이렇듯 IoT 디바이스는 사용자가 쉽게 접근할 수 있는 곳에 위치해 있지만 다른 관점에서 바라보면 공격자 또한 디바이스에 접근할 가능성을 제공한다. 따라서 공격자의 물리적 접근으로 인해 IoT 디바이스는 새로운 보안 취약점을 갖는다.

[0003] 공격자가 디바이스의 접근성을 악용하여 취할 수 있는 행위로는 펌웨어 추출이 있다. 공격자는 디바이스 내 MCU(Micro-Controller Unit) 칩을 추출하여 내부 펌웨어의 바이너리를 분석하여 서비스 어플리케이션의 제어 흐름을 파악할 수 있다. 공격자는 분석한 펌웨어를 기반으로 다양한 보안 위협을 수행할 수 있으며 대표적인 위협으로는 코드 재사용 공격(Code Reuse Attack)이 있다.

[0004] 기존 보안 솔루션은 데이터의 암/복호화를 통한 안전한 통신과 인증 및 검증 시스템을 도입하여 외부 위협으로부터 디바이스를 보호하는 데에 초점을 두었다. 하지만 코드 재사용 공격은 플래시 메모리에 저장된 적법한 펌웨어 바이너리만을 이용해 새로운 프로그래밍을 구현하기 때문에 기존 보안 솔루션만으로는 이를 해결할 수 없

었다.

[0005] 최근 CFI(Control Flow Integrity) 관련 연구들이 꾸준히 진행되어 코드 재사용 공격을 대비하고 있다. CFI(Control Flow Integrity)는 소프트웨어 실행이 사전에 CFG(Control Flow Graph)의 경로대로 흘러가게 지시하는 보안 정책을 의미한다. 그러나 저사양 IoT 디바이스라는 환경에서 하드웨어 기반의 암호화 방식은 제조 원가 비용을 증가시키고 여전히 펌웨어 추출 공격을 통해 해당 보호 기법에 취약하다. 공격자가 펌웨어를 추출하더라도 코드 재사용 공격을 막을 수 있을 뿐만 아니라 저사양 IoT 디바이스에 적합한 보안책을 마련할 필요가 있다.

[0006] 본 발명은 상기와 같은 코드 재사용 공격을 막기 위하여 반환 명령어 난독화 방법 및 난독화된 반환 명령어 실행방법을 제시하고자 한다.

선행기술문헌

특허문헌

[0007] (특허문헌 0001) [특허문헌 1] 한국공개특허공보 제10-2017-0055933호

발명의 내용

해결하려는 과제

[0008] 본 발명은 전술한 문제점을 해결하기 위하여 창출된 것으로, 초기화(Initialization) 모듈을 이용하여 반환 명령어를 난독화하는 방법 및 반환 제어 흐름(Return Control Flow) 모듈을 이용하여 난독화된 반환 명령어를 실행하는 방법을 제공하는 것을 그 목적으로 한다.

[0009] 본 발명의 목적들은 이상에서 언급한 목적들로 제한되지 않으며, 언급되지 않은 또 다른 목적들은 아래의 기재로부터 명확하게 이해될 수 있을 것이다.

과제의 해결 수단

[0010] 상기한 목적들을 달성하기 위하여, 본 발명의 일 실시예에 따른 반환 명령어 난독화 방법 및 난독화된 반환 명령어 실행방법이 개시된다. 상기 방법은 컨트롤러가 수행하는 펌웨어 내 반환 명령어 난독화 방법에 있어서, 초기화(Initialization) 모듈을 통해 디바이스 내 컨트롤러(MCU)와 별도로 존재하는 보안칩(Secure Element: SE)으로부터 암호키를 수신하는 단계 및 암호키 수신하는 단계 이후, 초기화(Initialization) 모듈을 통해 컨트롤러(MCU)에 포함된 플래시 메모리에 존재하는 암호화된 반환 명령어의 어드레스를 탐색하는 스캐닝 단계를 포함할 수 있다.

[0011] 또한, 본 발명의 일 실시예에 따르면, 스캐닝 단계에서 탐색한 어드레스를 수집하고, 초기화(Initialization) 모듈을 통해 컨트롤러(MCU)에 포함된 정적 메모리(SRAM)내에 인덱스화하여 저장하는 어드레스 저장 단계를 더 포함할 수 있다.

[0012] 또한, 본 발명의 일 실시예에 따르면, 암호키를 수신하는 단계 이후, 수신한 암호키를 컨트롤러에 포함된 정적 메모리(SRAM)에 저장하는 단계를 더 포함할 수 있다.

[0013] 또한, 본 발명의 일 실시예에 따르면, 초기화(Initialization) 모듈은 코드 인스트루먼테이션 과정을 통해 소스 바이너리 파일에 삽입되며, 플래시 메모리에 독립적으로 존재하는 모듈일 수 있다.

[0014] 또한, 본 발명의 일 실시예에 따르면, 반환 명령어 난독화 방법에 의해 난독화된 반환 명령어를 컨트롤러에서 실행하는 방법에 있어서, 플래시 메모리에 저장된 펌웨어를 실행할 때, 정적 메모리(SRAM)에 저장된 암호키를 이용하여 암호화된 반환 명령어를 복호화하고, 복호화된 반환 명령어를 실행하는 실행(Runtime) 단계를 포함할 수 있다.

[0015] 또한, 본 발명의 일 실시예에 따르면, 실행(Runtime) 단계는 컨트롤러(MCU)가 암호화된 반환 명령어를 실행하기 이전에 반환 제어 흐름(Return Control Flow) 모듈을 실행하는 단계를 더 포함할 수 있다.

[0016] 또한, 본 발명의 일 실시예에 따르면, 반환 제어 흐름(Return Control Flow) 모듈은 코드 인스트루먼테이션 과

정을 통해 소스 바이너리 파일에 삽입되며, 암호화된 반환 명령어를 복호화하여 실행하는 모듈이며, 컨트롤러가 펌웨어 실행하여, 반환 제어 흐름(Return Control Flow) 모듈이 호출신호를 받으면, 정적 메모리(SRAM)에 저장된 어드레스를 로드하는 단계, 암호화된 반환 명령어를 암호키를 이용하여 복호화하며, 복호화된 명령어를 정적 메모리(SRAM)에 저장하는 단계 및 정적 메모리(SRAM)에 저장된 복호화된 명령어를 실행하는 단계를 포함할 수 있다.

[0017] 상기한 목적들을 달성하기 위한 구체적인 사항들은 첨부된 도면과 함께 상세하게 후술될 실시예들을 참조하면 명확해질 것이다.

[0018] 그러나, 본 발명은 이하에서 개시되는 실시예들에 한정되는 것이 아니라, 서로 다른 다양한 형태로 구성될 수 있으며, 본 발명의 개시가 완전하도록 하고 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자(이하, "통상의 기술자")에게 발명의 범주를 완전하게 알려주기 위해서 제공되는 것이다.

발명의 효과

[0019] 본 발명의 일 실시예에 의하면, 디바이스 내 플래시 메모리에 펌웨어를 저장할 때, 반환 명령어를 분기 명령을 가지는 일부 세그먼트만을 암호화하여 저장함으로써, 저사양 디바이스에서도 구동이 가능해진다.

[0020] 또한, 코드 재사용 공격에 활용되는 일부 명령어만 암호화함으로써 전체 암호화로 인해 유발되는 디바이스의 성능 저하를 해결할 뿐만 아니라 보안성까지 강화시킬 수 있다.

[0021] 또한, 디바이스 내의 별도로 존재하는 보안성이 강화된 하드웨어 보안칩(Secure Element: SE)을 삽입하여, 암호키를 별도로 보관함으로써 보안을 보다 강화시킬 수 있다.

[0022] 또한, 펌웨어 실행 시 복호화하는 과정에서 암호키의 저장은 휘발성이 있는 정적 메모리(SRAM)에 함으로써 보안을 강화 시킬 수 있다.

[0023] 상기의 방법을 통해 코드 재사용 공격의 일종인 반환 지향 프로그래밍을 방지할 수 있다.

[0024] 본 발명의 효과들은 상술된 효과들로 제한되지 않으며, 본 발명의 기술적 특징들에 의하여 기대되는 잠정적인 효과들은 아래의 기재로부터 명확하게 이해될 수 있을 것이다.

도면의 간단한 설명

[0025] 상기 언급된 본 발명 내용의 특징들이 상세하게, 보다 구체화된 설명으로, 이하의 실시예들을 참조하여 이해될 수 있도록, 실시예들 중 일부는 첨부되는 도면에서 도시된다. 또한, 도면과의 유사한 참조번호는 여러 측면에 걸쳐서 동일하거나 유사한 기능을 지칭하는 것으로 의도된다. 그러나 첨부된 도면들은 단지 본 발명 내용의 특정한 전형적인 실시예들만을 도시하는 것일 뿐, 본 발명의 범위를 한정하는 것으로 고려되지는 않으며, 동일한 효과를 갖는 다른 실시예들이 충분히 인식될 수 있다는 점을 유의하도록 한다.

도 1은 본 발명의 일 실시예에 따른 반환 명령어 난독화 방법을 도시한 순서도이다.

도 2는 본 발명의 일 실시예에 따른 반환 제어 흐름(Return Control Flow) 모듈의 과정을 도시한 순서도이다.

도 3은 본 발명의 일 실시예에 따른 반환 명령어 난독화 기법의 시스템 아키텍처를 도시한 도면이다.

도 4는 본 발명의 일 실시예에 따른 초기화(Initialization) 모듈을 통한 초기화(Initialization) 과정을 도시한 도면이다.

도 5는 본 발명의 일 실시예에 따른 반환 제어 흐름(Return Control Flow) 모듈이 포함된 실행(Runtime) 과정을 도시한 도면이다.

발명을 실시하기 위한 구체적인 내용

[0026] 본 발명은 다양한 변경을 가할 수 있고, 여러 가지 실시예들을 가질 수 있는 바, 특정 실시예들을 도면에 예시하고 이를 상세히 설명하고자 한다.

[0027] 청구범위에 개시된 발명의 다양한 특징들은 도면 및 상세한 설명을 고려하여 더 잘 이해될 수 있을 것이다. 명세서에 개시된 장치, 방법, 제법 및 다양한 실시예들은 예시를 위해서 제공되는 것이다. 개시된 구조 및 기능상의 특징들은 통상의 기술자로 하여금 다양한 실시예들을 구체적으로 실시할 수 있도록 하기 위한 것이고, 발명의 범위를 제한하기 위한 것이 아니다. 개시된 용어 및 문장들은 개시된 발명의 다양한 특징들을 이해하기 쉽게

설명하기 위한 것이고, 발명의 범위를 제한하기 위한 것이 아니다.

[0028] 본 발명을 설명함에 있어서, 관련된 공지기술에 대한 구체적인 설명이 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우, 그 상세한 설명을 생략한다.

[0029] 이하, 본 발명의 일 실시예에 따른 반환 명령어 난독화 방법 및 난독화된 반환 명령어 실행방법을 설명한다.

[0030] 도 1은 본 발명의 일 실시예에 따른 반환 명령어 난독화 방법을 도시한 순서도이다.

[0031] 도 1을 참조하면, 본 발명의 펌웨어 내 반환 명령어 난독화 방법(S10)에 있어서, 초기화(Initialization) 모듈을 통해 디바이스 내 컨트롤러(MCU)와 별도로 존재하는 보안칩(Secure Element: SE)으로부터 암호키를 수신하는 단계(S11), 암호키를 수신하는 단계 이후, 초기화(Initialization) 모듈을 통해 컨트롤러(MCU)에 포함된 플래시 메모리에 존재하는 암호화된 반환 명령어의 어드레스를 탐색하는 스캐닝 단계(S13) 및 스캐닝 단계에서 탐색한 어드레스를 수집하고, 초기화(Initialization) 모듈을 통해 컨트롤러(MCU)에 포함된 정적 메모리(SRAM)내에 인덱스화하여 저장하는 어드레스 저장 단계(S15)를 포함할 수 있다.

[0032] 일 실시예에서, 반환 명령어 난독화 방법(S10)은 초기화(Initialization) 모듈을 통해 디바이스 내 컨트롤러(MCU)와 별도로 존재하는 보안칩(Secure Element: SE)으로부터 암호키를 수신하는 단계(S11)를 포함할 수 있다.

[0033] 보다 구체적으로, 암호키를 수신하는 단계(S11)는 컨트롤러(MCU)가 플래시 메모리에 삽입된 초기화(Initialization) 모듈을 이용하여 보안칩(Secure Element: SE)으로부터 암호키를 수신할 수 있다. 암호키를 수신하는 단계 이후, 수신한 암호키를 컨트롤러 내에 포함된 정적 메모리(SRAM)에 저장하는 단계를 포함할 수 있다.

[0034] 일 실시예에서, 반환 명령어 난독화 방법(S10)은 암호키 수신하는 단계 이후, 상기 초기화(Initialization) 모듈을 통해 컨트롤러(MCU)에 포함된 플래시 메모리에 내 코드 세그먼트에 존재하는 암호화된 반환 명령어의 어드레스를 탐색하는 스캐닝 단계(S13)를 포함할 수 있다. 컨트롤러(MCU)는 어드레스 탐색을 위해서 암호화된 반환 명령어의 연산 부호값을 비교하여 주소 번지를 파악할 수 있다.

[0035] 일 실시예에서, 반환 명령어 난독화 방법(S10)은 스캐닝 단계에서 탐색한 어드레스를 수집하고, 초기화(Initialization) 모듈을 통해 컨트롤러(MCU)에 포함된 정적 메모리(SRAM)내에 인덱스화하여 저장하는 어드레스 저장 단계(S15)를 포함할 수 있다. 추후 펌웨어 실행 시, 복호화 과정에서 사용하기 위한 용도로, 정적 메모리(SRAM)내 리턴 테이블에 인덱스화하여 저장할 수 있다.

[0036] 코드 세그먼트는 소스 바이너리 파일 중에서 분기 명령을 포함하는 코드일 수 있다. 코드 전체를 암호화하는 것보다 분기 명령을 가지는 코드만 암호화함으로써 저사양 IoT 디바이스에서도 구동이 가능하며, 보안을 강화시킬 수 있다.

[0037] 하드웨어 보안칩(SE)(210)은 디바이스 내에서 컨트롤러(MCU)와 별도로 존재하는 하드웨어 보안칩(SE)으로써, 보안키를 별도의 하드웨어 보안칩(SE)에 보관함으로써 암호키의 유출의 위험성을 줄일 수 있다.

[0038] 초기화(Initialization) 모듈은 코드 인스트루먼테이션 과정을 통해 소스 바이너리 파일에 삽입되며, 플래시 메모리에 독립적으로 존재하는 모듈일 수 있다.

[0039] 도 2는 본 발명의 일 실시예에 따른 반환 제어 흐름(Return Control Flow) 모듈의 과정을 도시한 순서도이다.

[0040] 도 2를 참조하면, 반환 명령어 난독화 방법에 의해 난독화된 반환 명령어를 실행하는 방법에 있어서, 플래시 메모리에 기저장된 펌웨어를 실행할 때, 정적 메모리(SRAM)에 저장된 암호키를 이용하여 암호화된 반환 명령어를 복호화하고, 복호화된 반환 명령어를 실행하는 실행(Runtime) 단계를 포함할 수 있다. 실행(Runtime) 단계는, 컨트롤러(MCU)가 암호화된 반환 명령어를 실행하기 이전에 반환 제어 흐름(Return Control Flow) 모듈(S20)을 실행하는 단계를 더 포함할 수 있다.

[0041] 보다 구체적으로, 실행(Runtime) 단계는 코드 세그먼트에 마지막 명령어가 암호화된 형태로 저장되며, 컨트롤러(MCU)가 코드 세그먼트에 암호화된 반환 명령어를 실행하기 이전에 반환 제어 흐름(Return Control Flow) 모듈(S20)을 실행할 수 있다. 반환 제어 흐름(Return Control Flow) 모듈(S20)은 소스 바이너리 파일에 삽입되어 암호화된 반환 명령어를 복호화하여 실행하는 모듈이며, 컨트롤러가 펌웨어 실행하여, 반환 제어 흐름(Return Control Flow) 모듈이 호출신호를 받으면, 컨트롤러(MCU)는 디바이스 내 정적 메모리(SRAM)에 저장된 리턴 테이블을 참조하여 복호화 하려는 암호화된 반환 명령어의 어드레스를 로드하는 단계(S21), 로드한 암호화된 반환 명령어를 초기화(Initialization) 과정에서 수신한 암호키를 이용하여 복호화하며, 복호화된 명령어를 정적 메

모리(SRAM)에 저장하는 단계(S23) 및 정적 메모리(SRAM)에 저장된 복호화된 명령어를 실행하는 단계(S25)를 포함하는 모듈일 수 있다.

[0042] 도 3은 본 발명의 일 실시예에 따른 반환 명령어 난독화 기법의 시스템 아키텍처를 도시한 도면이다.

[0043] 도 3을 참조하면, 시스템 아키텍처(100)는 초기화(Initialization) 모듈과 반환 제어 흐름(Return Control Flow) 모듈의 두 개의 모듈을 원본 소스 바이너리 파일에 삽입하는 코드 인스트루먼테이션 과정을 통해 인스트루먼테이션된 소스파일을 생성할 수 있다. 이후 반환 명령어를 암호화한 상태로 컨트롤러(MCU) 내 플래시 메모리 영역에 재작성하는 바이너리 수정 과정을 수행하여 최종적으로 재작성된 바이너리 파일을 생성할 수 있다.

[0044] 코드 인스트루먼테이션은 실행 파일의 오류를 진단하거나 추적 정보를 쓰기 위해 제품의 성능 정도를 모니터하거나 측정하는 기능을 말한다. 시스템에서 특정 구성 요소를 모니터링하는 코드 명령어 형태로 인스트러먼테이션을 구현할 수 있다. 그러나 IoT 디바이스는 대부분이 소형화, 저전력이라는 제약된 환경으로써 운영체제가 없는 형태로 제작되기 때문에 백도어를 활용하여 시스템의 오류를 체크하는 dynamic instrumentation 방식이 힘들다. 따라서, 본 특허에서는 코드 인스트러먼테이션을 하기 위해 반환 명령어가 실행되는 시점과 메인함수가 실행되는 시점에 삽입된 모듈을 직접 삽입하는 static 코드 인스트러먼테이션 방식을 활용할 수 있다.

[0045] 링커 스크립터를 수정을 하여 삽입된 두 모듈은 분리된 섹션에 보관 및 관리할 수 있다. 따라서 링커 스크립터를 수정하여 독립된 영역을 생성하고 모듈의 소스 코드를 저장할 수 있다.

[0046] 바이너리 수정의 목적은 IoT 디바이스 내의 플래시 메모리에 pop과 같은 반환 명령어를 암호화된 상태로 저장하는 것이다. 앞서 설명한대로 저사양 IoT 디바이스는 OS가 없는 환경이 대부분이기 때문에 백도어를 활용한 명령어 체크가 불가능하다. 따라서 펌웨어 바이너리에 직접 체크를 위한 바이너리 코드를 삽입할 필요가 있다. 바이너리 수정을 통해 암호화된 명령어는 추후에 초기화(Initialization) 모듈과 반환 제어 흐름(Return Control Flow) 모듈에 의해 안전하게 복호화 및 실행될 수 있다.

[0047] 코드 인스트러먼테이션을 통해 삽입할 두 모듈은 암호화된 명령어 정보를 수집하여 그 주소 번지를 정적 메모리(SRAM) 영역 내 테이블에 저장하는 초기화(Initialization) 모듈과 펌웨어의 런타임(Runtime) 시 암호화된 명령어를 안전하게 실행하는 반환 제어 흐름(Return Control Flow) 모듈로 구성된다.

[0048] 초기화(Initialization) 모듈을 통한 초기화(Initialization) 과정에 대한 설명 및 반환 제어 흐름(Return Control Flow) 모듈이 포함된 실행(Runtime) 과정에 대한 설명은 도 4 및 도 5에 대한 설명에서 각각 구체적으로 후술한다.

[0049] 도 4는 본 발명의 일 실시예에 따른 초기화(Initialization) 모듈을 통한 초기화(Initialization) 과정을 도시한 도면이다.

[0050] 도 4를 참조하면, 초기화(Initialization) 과정은 컨트롤러(MCU)가 플래시 메모리 내에 존재하는 초기화(Initialization) 모듈을 통해 디바이스 내의 별도로 존재하는 하드웨어 보안칩(SE)으로부터 암호키를 수신하여 반환 명령어를 암호화시키는 암호키 수신 단계, 플래시 메모리 내의 코드 세그먼트에 존재하는 암호화된 반환 명령어의 주소를 탐색하는 스캐닝 단계 및 탐색한 주소를 수집하여 정적 메모리(SRAM)내 리턴 테이블에 인덱스화 하여 저장하는 주소 저장 단계를 포함할 수 있다.

[0051] 보다 구체적으로, 암호키 수신하는 단계는 초기화(Initialization) 모듈이 실행되면, 컨트롤러(MCU)는 플래시 메모리 내에 존재하는 초기화(Initialization) 모듈 및 디바이스 내에 별도로 존재하는 하드웨어 보안 칩(SE)과의 통신을 통해 암호키를 수신하는 단계일 수 있다. 수신한 암호키는 AES128 암호 알고리즘을 활용하여 16바이트의 데이터 길이를 갖는다. 초기화(Initialization) 모듈과 하드웨어 보안칩(SE) 간의 교환메시지는 암호화된 상태로 통신하고, 암호화에 사용되는 암호키는 안전하게 생성하여 초기화(Initialization) 모듈과 하드웨어 보안칩(SE) 내에 저장될 수 있다.

[0052] 또한, 스캐닝 단계는 암호키를 수신하는 단계를 통해 암호키 수신이 완료되면, 플래시 메모리 내 코드 세그먼트에 존재하는 암호화된 반환 명령어의 주소를 탐색하는 단계일 수 있다. 탐색을 위해서 반환 명령어의 연산 부호값을 비교하여 주소 번지를 파악할 수 있다.

[0053] 또한, 주소 저장 단계는 스캐닝 단계에서 탐색한 반환 명령어의 주소를 정적 메모리(SRAM) 내 리턴 테이블에 인덱스화하여 저장하는 단계일 수 있다. 이 테이블은 나중에 복화화를 하기 위한 용도로 활용될 수 있다. 이 리턴 테이블은 휘발성 메모리인 정적 메모리(SRAM) 영역에 존재하기 때문에 디바이스 부트 시, 그 흔적을 남기지 않

게 되어 단순 펌웨어 추출만으로는 분석이 불가능 할 수 있다.

[0054] 도 5는 본 발명의 일 실시예에 따른 반환 제어 흐름(Return Control Flow) 모듈이 포함된 실행(Runtime) 과정을 도시한 도면이다.

[0055] 도 5를 참조하면, 실행(Runtime) 단계는 코드 세그먼트에 마지막 명령어가 암호화된 형태로 저장되며, 컨트롤러(MCU)가 코드 세그먼트에 암호화된 반환 명령어를 실행하기 이전에 반환 제어 흐름(Return Control Flow) 모듈을 실행할 수 있다. 반환 제어 흐름(Return Control Flow) 모듈(S20)은 소스 바이너리 파일에 삽입되어 암호화된 반환 명령어를 복호화하여 실행하는 모듈이며, 컨트롤러가 펌웨어 실행하여, 반환 제어 흐름(Return Control Flow) 모듈(S20)이 호출신호를 받으면, 컨트롤러(MCU)는 디바이스 내 정적 메모리(SRAM)에 저장된 리턴 테이블을 참조하여 복호화 하려는 암호화된 반환 명령어의 주소를 로드하는 단계(S21), 로드한 암호화된 반환 명령어를 초기화(Initialization) 단계에서 수신한 암호키를 이용하여 복호화하며, 복호화된 명령어를 정적 메모리(SRAM)에 저장하는 단계(S23) 및 정적 메모리(SRAM)에 저장된 복호화된 명령어를 실행하는 단계(S25)를 포함하는 모듈일 수 있다.

[0056] 보다 구체적으로, 도 5의 원본 코드 세그먼트의 경우, 원본 소소 코드에서 foo 함수는 일련의 명령어가 모두 수행되고 마지막 명령어인 pop 명령어가 나오면, 이전에 호출된 func 함수로 제어 흐림이 변경된다.

[0057] 반면에, 재작성(re-write)한 코드 세그먼트의 경우는, 마지막 명령어인 pop 명령어가 암호화된 형태(enc_ins)로 저장되어 있으며, 소스 코드에서 foo 함수의 명령어가 모두 수행되고 암호화된 pop 명령어(enc_ins)가 나오기 이전에 반환 제어 흐름(Return Control Flow) 모듈이 삽입되어 있다. 또한, 암호화된 pop명령어(enc_ins)가 실행되기 이전에 반환 제어 흐름(Return Control Flow) 모듈을 실행하는 명령어를 추가함으로써 하기와 같은 동작들이 순차적으로 진행될 수 있다.

[0058] 또한, 도 5를 참조하면, 실선으로 표시된 direct call은 단순히 이전 명령어 다음 순서의 명령어를 의미하므로 보호의 대상이 아니며, 파선으로 표시된 indirect call은 이전의 명령어를 참조해서 다음 순서의 명령어로 이동을 하는 것으로 보호의 필요성이 있는 명령어이다.

[0059] 예를 들어, foo 함수에서 암호화된 pop명령어(enc_ins)가 나오기 이전에 개발자가 작성 및 삽입한 “bEx_Ret_CF”는 반환 제어 흐름(Return Control Flow) 모듈을 호출할 수 있다. 호출된 반환 제어 흐름(Return Control Flow) 모듈은 기저장된 명령어들을 수행할 수 있다. 반환 제어 흐름(Return Control Flow) 모듈은 암호화된 반환 명령어의 주소 로드(load enc_ins), 복호화(decryption) 및 실행(execution)의 명령어를 포함할 수 있다. 암호화된 반환 명령어의 주소 로드(load enc_ins)는 반환 제어 흐름 모듈로 진입하면 정적 메모리(SRAM)에 저장된 리턴 테이블을 참조하여 현재 복호화 하려는 암호화된 명령어의 주소 번지를 로드할 수 있다. 복호화(decryption)는 로드한 enc_ins는 초기화(Initialization) 모듈에서 수신한 암호키로 복호화 할 수 있다. 실행(execution)은 정적 메모리(SRAM)에 저장된 복호화된 명령어를 실행하여 func 함수로 반환할 수 있다. 위와 같은 일련의 작업들을 통해 코드 세그먼트에 저장된 모든 반환 명령어를 난독화하고, 반환 제어 흐름(Return Control Flow) 모듈은 명령어를 실행할 때만 원본으로 복호화하여 실행할 수 있다. 상기 예시는 본 개시를 설명하기 위한 예시일 뿐, 본 개시는 이에 제한되지 않는다.

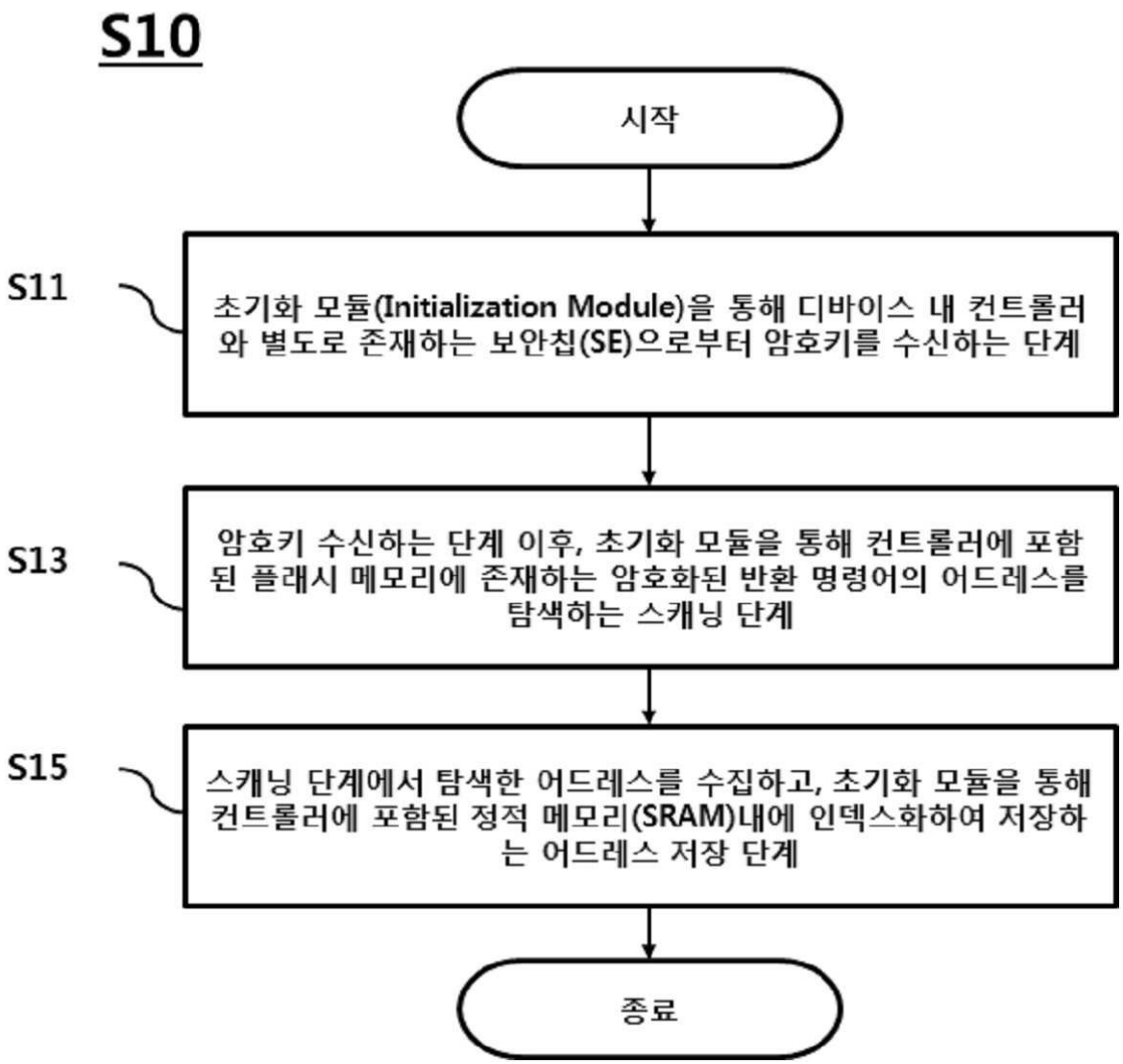
[0060] 이상의 설명은 본 발명의 기술적 사상을 예시적으로 설명한 것에 불과한 것으로, 통상의 기술자라면 본 발명의 본질적인 특성이 벗어나지 않는 범위에서 다양한 변경 및 수정이 가능할 것이다.

[0061] 따라서, 본 명세서에 개시된 실시예들은 본 발명의 기술적 사상을 한정하기 위한 것이 아니라, 설명하기 위한 것이고, 이러한 실시예들에 의하여 본 발명의 범위가 한정되는 것은 아니다.

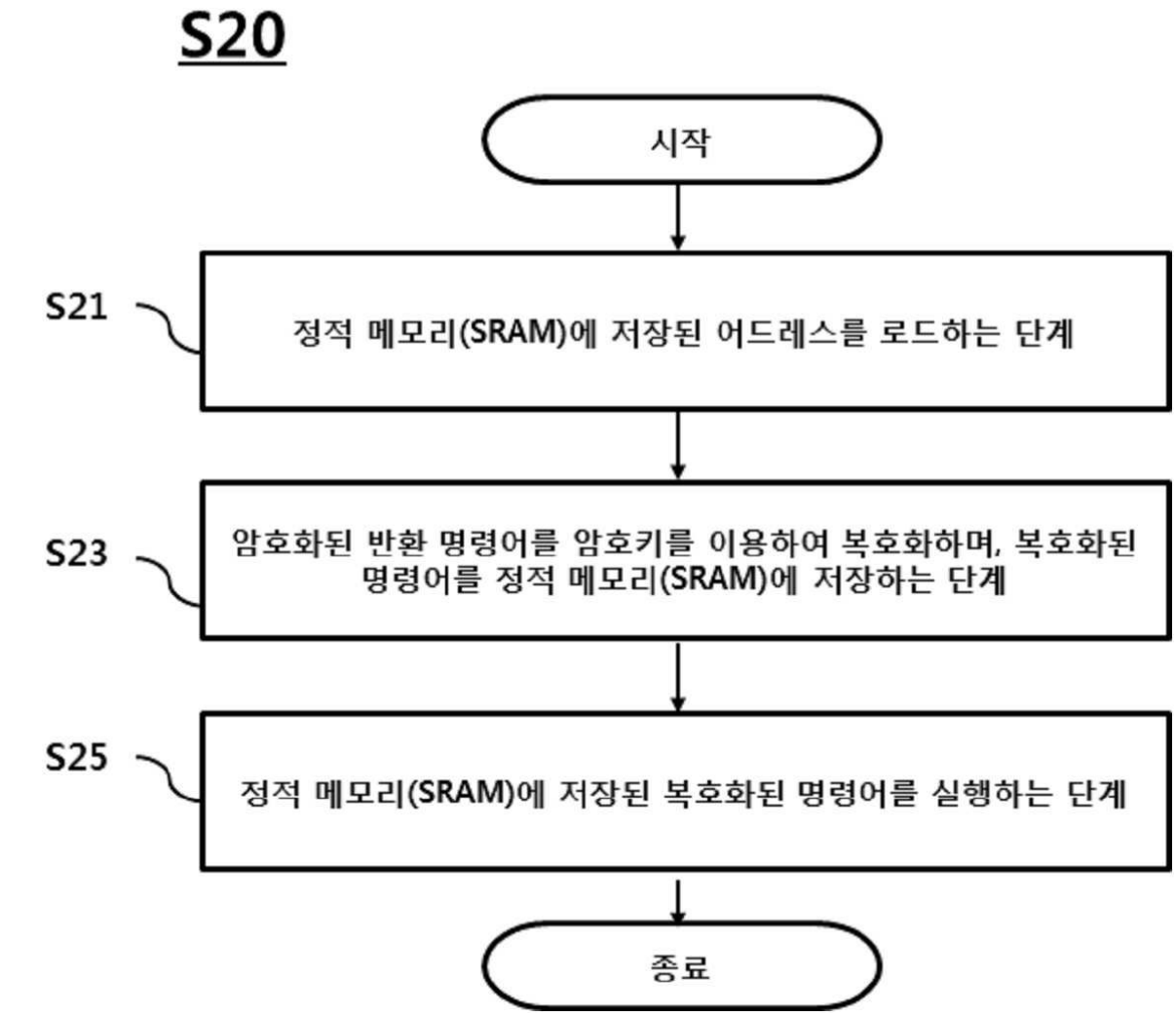
[0062] 본 발명의 보호범위는 청구범위에 의하여 해석되어야 하며, 그와 동등한 범위 내에 있는 모든 기술 사상은 본 발명의 권리범위에 포함되는 것으로 이해되어야 한다.

도면

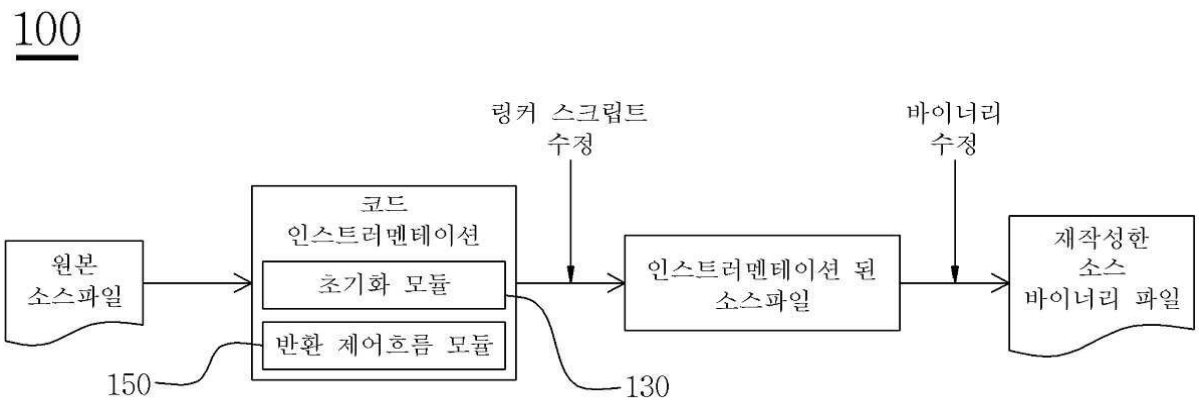
도면1


S10
시작
S11
초기화 모듈(Initialization Module)을 통해 디바이스 내 컨트롤러와 별도로 존재하는 보안칩(SE)으로부터 암호키를 수신하는 단계
S13
암호키 수신하는 단계 이후, 초기화 모듈을 통해 컨트롤러에 포함된 플래시 메모리에 존재하는 암호화된 반환 명령어의 어드레스를 탐색하는 스캐닝 단계
S15
스캐닝 단계에서 탐색한 어드레스를 수집하고, 초기화 모듈을 통해 컨트롤러에 포함된 정적 메모리(SRAM)내에 인덱스화하여 저장하는 어드레스 저장 단계
종료

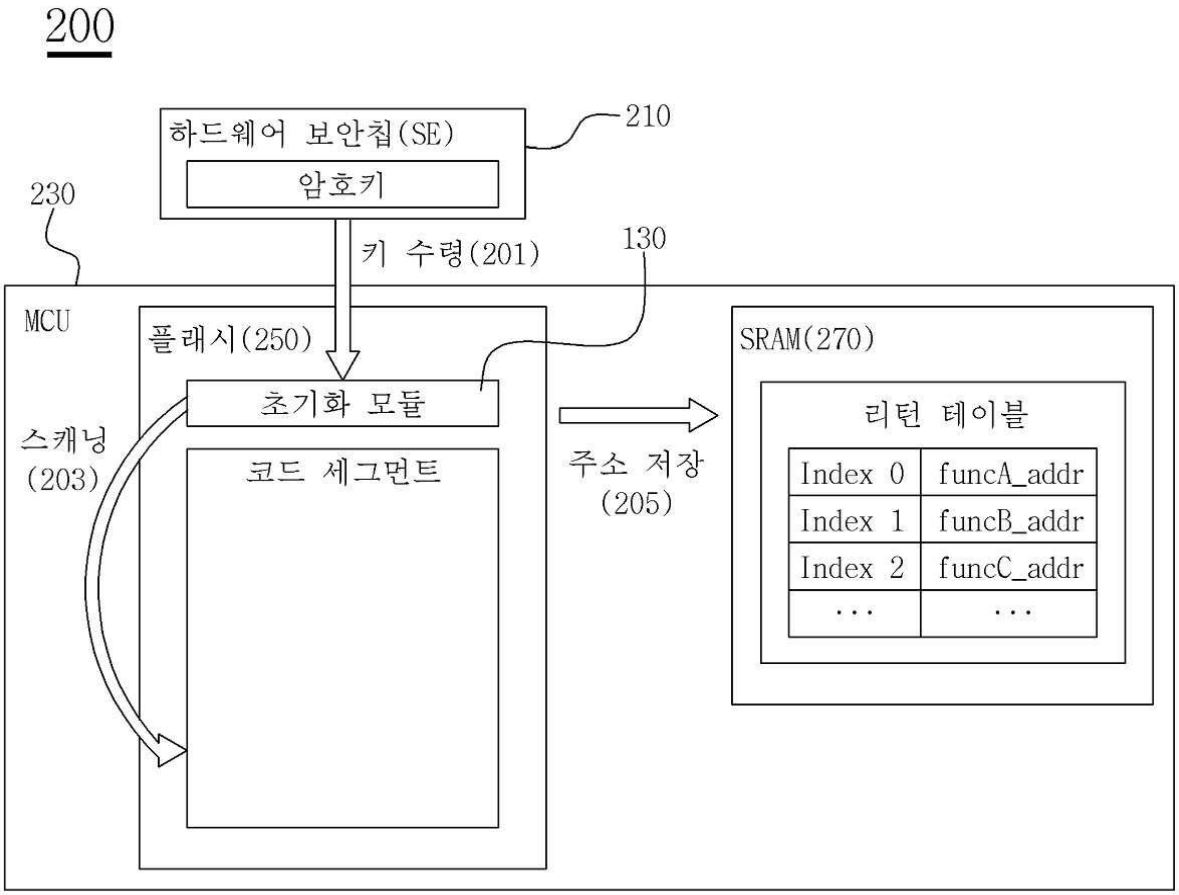
도면2


S20
시작
S21
정적 메모리(SRAM)에 저장된 어드레스를 로드하는 단계
S23
암호화된 반환 명령어를 암호키를 이용하여 복호화하며, 복호화된 명령어를 정적 메모리(SRAM)에 저장하는 단계
S25
정적 메모리(SRAM)에 저장된 복호화된 명령어를 실행하는 단계
종료

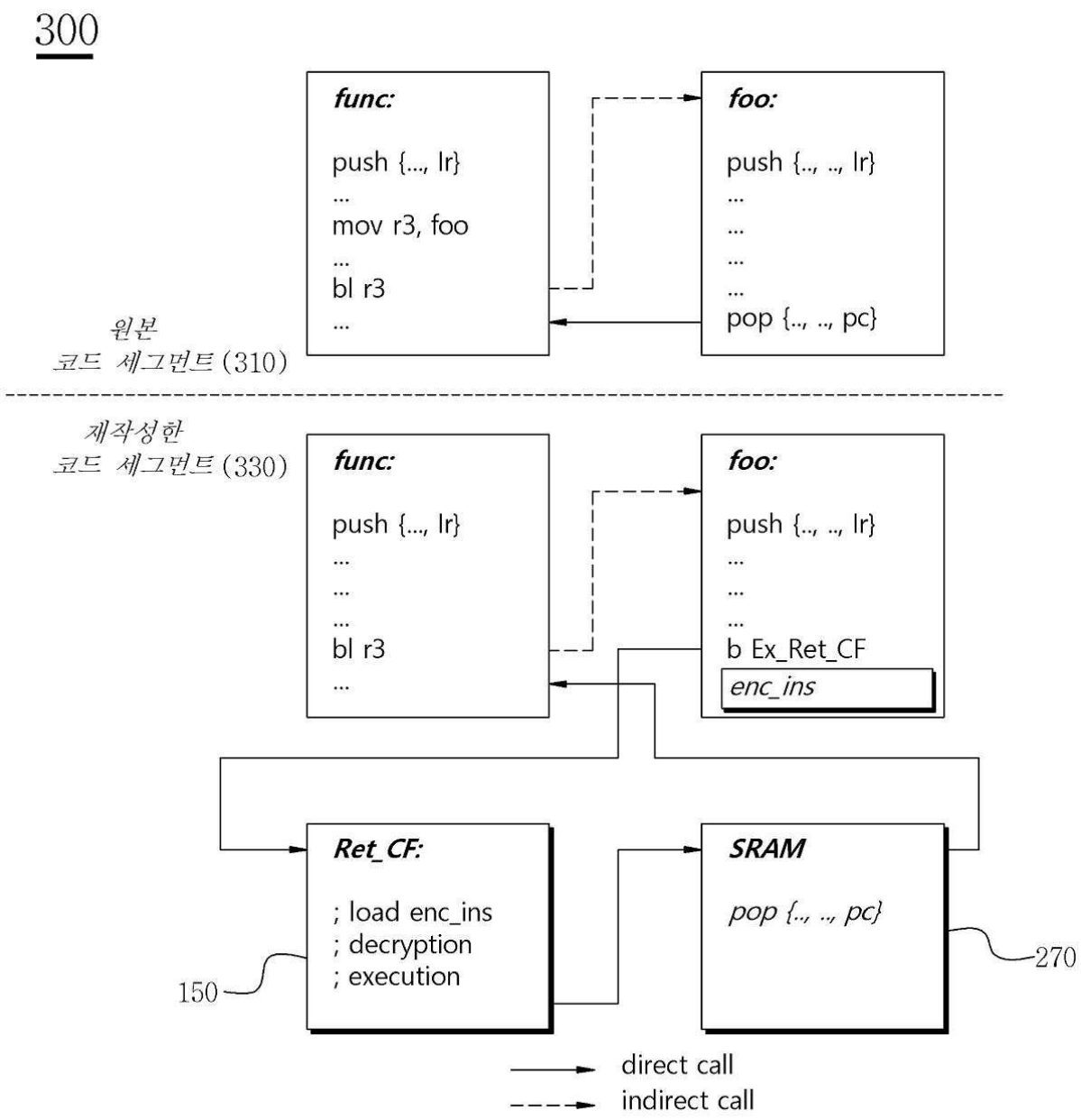
도면3


100
링커 스크립트 수정
바이너리 수정
원본 소스파일
코드 인스트러멘테이션
초기화 모듈
반환 제어흐름 모듈
150
130
인스트러멘테이션 된 소스파일
재작성한 소스 바이너리 파일

도면4


200
하드웨어 보안칩(SE)
암호키
210
230
키 수령(201)
130
MCU
플래시(250)
초기화 모듈
코드 세그먼트
스캐닝
(203)
주소 저장
(205)
SRAM(270)
리턴 테이블
Index 0
funcA_addr
Index 1
funcB_addr
Index 2
funcC_addr
...
...

도면5


300
func:
push {..., lr}
...
mov r3, foo
...
bl r3
...
foo:
push {.., .., lr}
...
...
...
...
pop {.., .., pc}
원본
코드 세그먼트 (310)
제작성한
코드 세그먼트 (330)
func:
push {..., lr}
...
...
...
bl r3
...
foo:
push {.., .., lr}
...
...
...
b Ex_Ret_CF
enc_ins
Ret_CF:
; load enc_ins
; decryption
; execution
150
SRAM
pop {.., .., pc}
270
direct call
indirect call